

Chapter 2: Template files

The TRS process uses action and screen templates to maneuver through the screens of a host application. These templates exchange information with the host application screens and transfer information to and from the TRS's buffers. Coupled with the VT100 emulation software and hardware, they provide the host with exactly the same type of input as a terminal operator.

This chapter explains how to:

- Determine the actions a terminal operator performs to enter and retrieve information
- Create the action and initial action template files that define the sequence of host application screens for each transaction
- Create the screen template files that define the sequence of fields encountered on each screen

Note: If you make backups of your template files, do not store them in the `/u/ivr/3270` directory or in any subdirectory under `/3270`. You should make a directory at the same hierarchical level or higher as `/3270`. For example, if `/u/ivr/3270bak` is specified, the TRS process searches the `/3270` directory and any subdirectories within it for files with the `.act` or `.scn` extensions.

Determining the required transactions

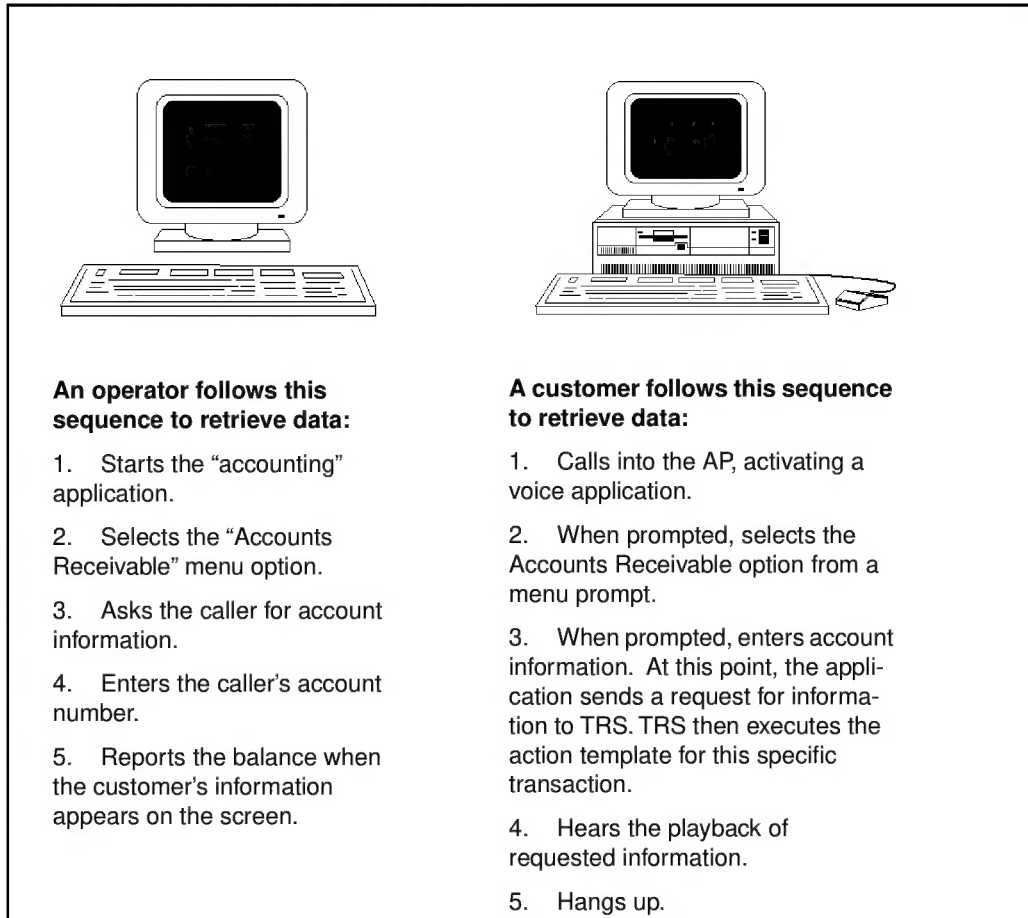
Imagine that you are an operator sitting at a terminal. In order to perform a specific task, you type information and press function keys until you accomplish the desired task. Perhaps a caller asks you to look up a customer's account balance, or enter an order. The series of steps you perform at the terminal enable the application on the host computer to complete the transaction.

To develop a voice application that accesses the same information as a terminal operator, you need to tell Meridian IVR 2.0/I how to execute the same series of actions that the terminal operator executes. You provide this information in ASCII files called template files.

Template files provide the layout and content of each screen in the host application as the terminal operator sees them. Figure 2-1 compares a transaction done by a terminal operator to one done by a customer calling into a voice response system.

Note: The first step performed by the terminal operator is not performed by the action template. It is performed by the initial-action template (described later in this chapter). The initial-action template handles the login and moves the application to the appropriate screen to begin the transaction.

Figure 2-1
Voice response system vs. terminal operator

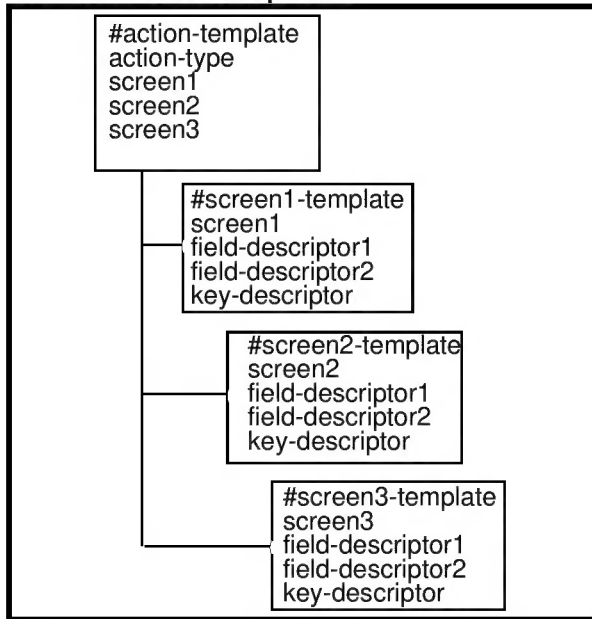


The TRS requires two types of template files:

- *Action templates*, which describe the sequence of screens traversed in order to perform a specific transaction.
- *Screen templates*, which validate each screen, define the fields on the screen that require data, and define all keystrokes required for the screen.

Figure 2-2 shows how screen templates relate to action templates.

Figure 2-2
Action and screen templates



VT100 based applications often have format inconsistencies that make it difficult for the TRS to efficiently determine when the host application is ready for input and what region of the host screen contains vital information. These inconsistencies are due to the character based nature of the VT100 protocol.

In addition, the VT100 communication protocol has no way of notifying the TRS that host data transmission has ended. From the terminal operator's perspective, it is easy to tell when host transmission ends because the operator's requested information appears on the screen. From the TRS's perspective, there is nothing inherent in the VT100 protocol to provide notification of the end of host output. The TRS encounters a stream of data from the host, and from this must determine the identity of each screen as well as locate the end each screen. To enable the TRS to do this, as well as cope with screen inconsistencies, you must create a file called **screen.conf**.

The action templates, screen templates, and screen.conf file are ASCII text files that use a simple syntax to define the screen flow and input/output fields. The sections that follow provide a detailed explanation of the templates, as well as the information necessary to create the **screen.conf** file.

Action templates

A VT100 transaction typically moves through several screens until it locates specific information. The screens may be a series of commands issued at the operating system prompt, or they may be screens within an application running on the host computer. Whenever Meridian IVR 2.0/I references an action template, the VT100 Gateway executes the screen templates listed in the action template, moving through the application just as a terminal operator would. An action template must specify the same sequence of screens that the terminal operator traverses.

A separate action template defines each transaction. In the example shown in Figure 2-1, if you want to select a menu option other than “Accounts Receivable,” you would define another action template.

Action templates describe the flow of the screens that comprise a particular transaction. For example, if you want a transaction to access billing information for a specific client, as a terminal operator you would perform the following steps:

Procedure 2-1

Accessing transaction information

- 1 Log on to the computer.
- 2 Start the “acct” application.
- 3 Select the “Accounts Receivable” menu.
- 4 When prompted, enter the client’s account number and press the Return key. A screen would appear displaying the client’s billing information.
- 5 Read the account information on the screen.

In a Meridian IVR 2.0/I VT100 transaction, an initial-action template performs steps 1 and 2. The initial-action template automatically executes when the TRS process starts (initial-action templates are described later in this chapter.). An action template created to execute this transaction would perform steps 3 through 5.

Action template syntax

An action template is an ASCII file created with a text editor. The action template files you create must reside in the `/u/ivr/3270` directory or in a subdirectory below `/u/ivr/3270`. They must also have the file name extension `.act`. For example, if you created an action template called **getbalance.act**, it would have this path:

```
u/ivr/3270/getbalance.act
```

The syntax of an action template is shown in Figure 2-3.

Figure 2-3
Action template syntax

```
#comment
action-name app-name reset-action logout-action <manual mode>
screen-template
screen-template
•
•
•
```

The lines depicted as • represent additional screen templates used in the transaction. Each screen template corresponds to a specific host application screen which appears on the terminal during a session. Screen templates are listed in the action template in the same order as they appear during the actual terminal session. (Screen templates are discussed later in this chapter). The example in Figure 2-4 illustrates an action template file which describes a transaction for retrieving account information. This sample application is used as a development example throughout this guide.

Figure 2-4
Action template for accounting application

```
#Example action template file: filename is getbalance.act
getbalance  accounting  reset_cust  logout_cust
#acctrec chooses the balance option from the main menu
acctrec
#acctno specifies the account number for the customer
acctno
#customer displays the customer's account balance
customer
```

In Figure 2-4, *action-name* is **getbalance**, the name of the action template file without the **.act** extension. The *app-name* is **accounting**. The *reset-action* is **reset_cust** (file name **reset_cust.act**), and the *logout-action* is **logout_cust** (file name **logout_cust.act**). *Manual mode* is omitted because manual mode is not required for this transaction (a description of manual mode follows).

The remaining lines identify the sequence of screens (**acctrec**, **acctno**, and **customer**) the TRS must traverse to retrieve the customer billing information. These screens are listed in the order that they must be accessed.

An explanation of each entry in the action template syntax follows.

#comment

The first line of the template in Figure 2-3 is a comment. The comment line is not required but is recommended to describe the purpose of the action template.

Comments start with the “#” symbol and can be embedded anywhere in the action template. If a comment begins a line, no non-comment fields may follow the comment in that line. However, a comment may appear after a non-comment field, such as after the screen template name as shown in Figure 2-4. It is good practice to heavily comment files so that changes can be made easily in the future.

action-name

The *action-name* is the file name of the action template file without the **.act** extension. The *action-name* is required to begin the transaction. For example, if the action template’s file name is **getbalance.act**, you would enter **getbalance** for the action-name.

app-name

When you set up your application processor with the VT100 Gateway, you must create a **trs.conf** file that assigns TRS session numbers to the application on the host computer (the **trs.conf** file is described in Chapter 3). Choose a name for the host computer application name; it does not need to match any actual application name on the host computer.

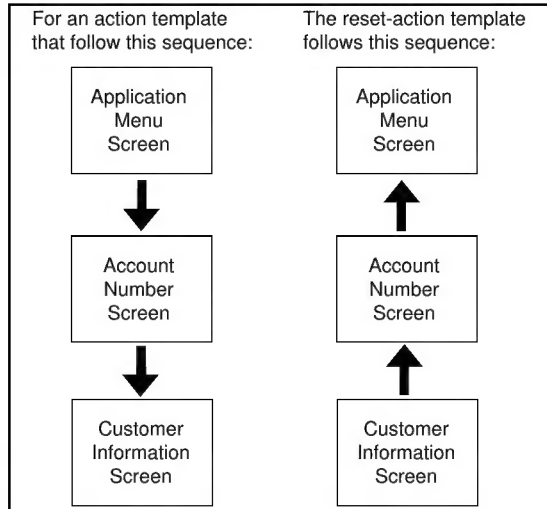
As an example, this guide uses the *app-name* “accounting” to represent the host computer “**acct**” application.

The *app-name* you specify in the action template must exist in the **trs.conf** file (discussed in Chapter 3). Meridian IVR 2.0/I passes the *app-name* to the TRS function, which uses the name to start the appropriate session with the host computer.

reset-action

The *reset-action* specifies an action template to be processed when the transaction completes or if the transaction fails. Typically, the reset-action template is used to bring the host computer application back to its main screen so it is ready to process the same type of transaction. Figure 2-5 shows the sequence a sample reset-action template follows.

Figure 2-5
Reset-action template sequence sample



Entering a hyphen "-" in the *reset-action* entry indicates that no reset-action template is specified.

If no reset-action template is specified and the transaction being executed by the action template fails, the logout-action template (described in the next section) is executed. If the transaction succeeds and there is no reset-action template specified, the host computer application remains at the screen where the transaction ended.

When you create a reset-action template, do not specify reset-action or logout-action templates in it. For example, Figure 2-6 shows a sample reset-action template.

Figure 2-6
Reset-action template sample

```
#This reset-action template returns the host computer application
#to the main menu screen from the customer information screen
#filename: reset_cust.act
reset_cust  accounting  -    -
clrcust
#exit the customer information screen
atmenu
#leave the session at the menu screen
```

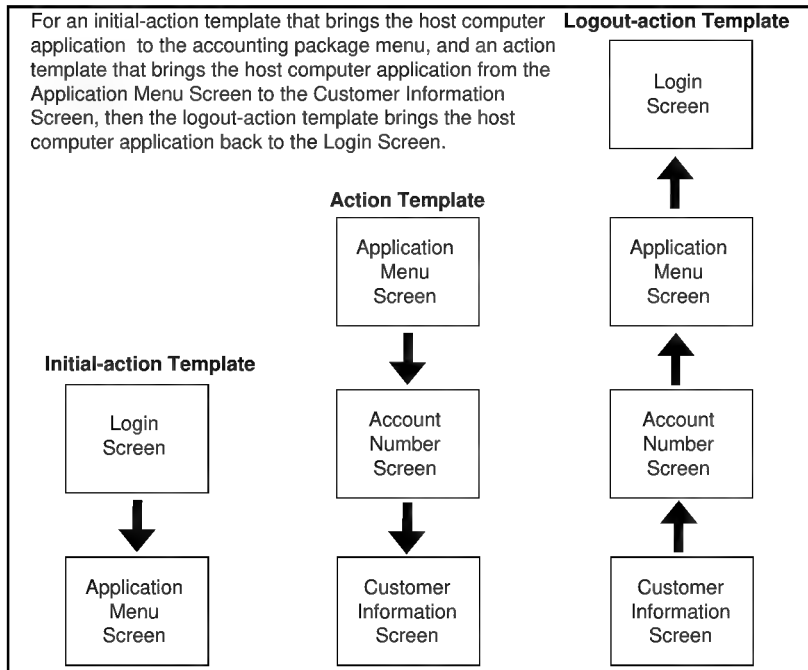
The action template using this reset-action template would enter **reset_cust** as the *reset-action* entry.

logout-action

The *logout-action* specifies a logout-action template to be executed if the reset-action template fails, or if the transaction fails and there is no reset-action template specified. If the transaction succeeds, the logout-action template is not executed.

The TRS uses the logout-action template to return the failed transaction to the initial screen (usually a login screen). After it successfully executes the logout-action template, it executes the initial-action template (described later in this chapter) after 30 seconds. Figure 2-7 shows the flow of the logout action template.

Figure 2-7
Logout action flow



The logout-action template locates the screen where the transaction failed. If for example, the transaction failed at the account number screen, the logout-action template locates the screen template with the appropriate validation tag and starts from that screen.

Entering a hyphen for the **logout-action** entry indicates that no logout-action template is specified. If neither a reset-action template nor a logout-action template is specified and the transaction fails, the host computer application remains at the point where the transaction failed. Future transactions that try to use this session would also experience errors because the screen where the session remained would not be the expected starting screen (unless the transaction can start from any screen).

When you create a logout-action template, do not specify reset-action or logout-action templates. Figure 2-8 shows a sample logout-action template.

Figure 2-8
Logout-action template sample

```
#This logout-action template returns the application to the login
#screen from the customer information screen
logout_cust  accounting  —  —
clrcust
#exits the customer information screen
clrmnu
#exits the acct application, shows the system prompt
```

screen-template

The *screen-template* (the file name of the template without the **.scn** extension) identifies the screen template used during the VT100 transaction. Enter the screen templates in the exact order they appear during the transaction. Each screen template must be listed on a separate line. The syntax for screen templates is described later in this chapter.

<manual mode> (optional field)

The **<manual-mode>** entry allows you to attach a session resource to a particular channel. You can then use the same session for consecutive Meridian IVR 2.0/I transactions. This type of session is not released when the transaction is finished. To exit manual mode you must execute a COMA cell in the Meridian IVR 2.0/I application or process another action template that does not contain the manual mode option. Chapter 4 describes how to use the COMA cell.

To specify manual mode, enter **manual** after the logout-action template name. If you omit **manual**, automatic mode is used for the template. In automatic mode, the session assigned to the action template is free for use when the transaction is completed.

Note: You should not specify a reset-action template in an action template that uses manual mode. Manual mode is designed to stay at a specific screen. The next transaction received by the session will start at the last screen of the action template that used manual mode. This next transaction *should* use automatic mode and specify a reset-action template that brings the session back to the starting screen of the first action template (that specified manual mode).

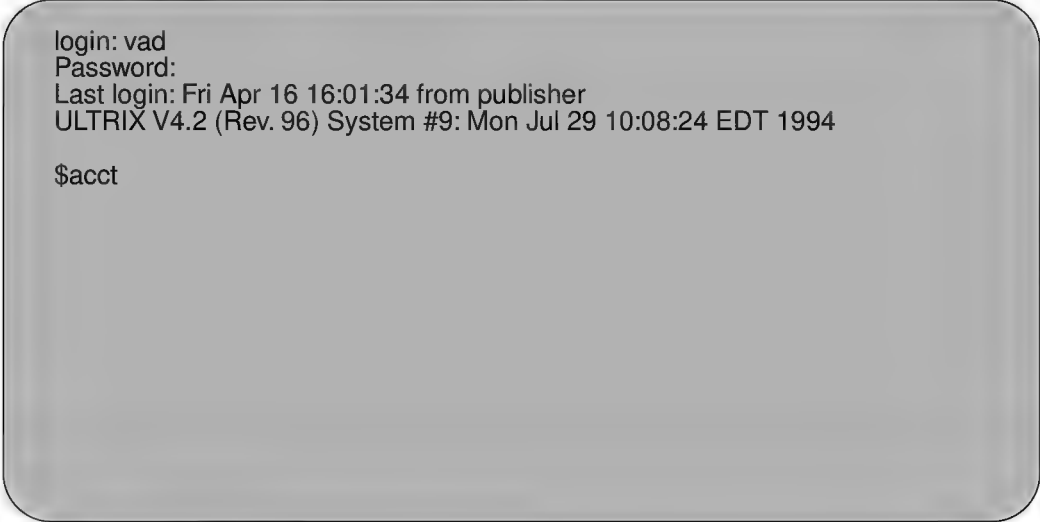
Screen templates

Screens used by the host computer could be a series of commands entered at the system prompt coupled with the system's response to those commands, or screens defined by applications on the host computer. You should define screen templates that issue the system commands to start the application (usually as part of the initial-action template), and then screen templates that make menu selections and enter or retrieve data from the screens displayed by the application. This guide uses an accounting application as an example.

Each screen contains fields. For the VT100 Gateway, a field is any place on the screen where data is entered or displayed. For example, the cursor location after a system prompt where you would type a command is considered a field. Also, within an application, the area on the screen where an account balance is displayed is also considered a field (the traditional definition of a field).

For a specific transaction, specific data is entered in certain fields, and data is read from other fields. A screen template identifies those fields on the screen that are used to process a transaction. Only the fields that are used in the transaction are included in the template. Figure 2-9 shows a sample screen.

Figure 2-9
Screen showing fields and the system prompt



```
login: vad
Password:
Last login: Fri Apr 16 16:01:34 from publisher
ULTRIX V4.2 (Rev. 96) System #9: Mon Jul 29 10:08:24 EDT 1994

$acct
```

In Figure 2-9, **vad** has been entered into the login field. If the **Return key** is pressed, the application starts and the screen is replaced by the application screen.

A sample application screen showing fields is shown in Figure 2-10. Here, the customer's name is entered in the customer field.

Figure 2-10
Application screen for accounting application

Account Number:845-23-87
Customer:Jane K. SmithCurrent Balance:2482.14
Address:19 Alpha RoadPayment Due:150.00
ChelmsfordPayment Due Date:4/30/93
MA 01824

Options:

1Print invoice
2Enter payment
3Enter purchase
4Exit

Type menu selection:

Different transactions may access different fields on a screen. For example, a transaction to locate the payment due would only need to access that field, while a transaction retrieving the customer's balance would only need to access the Current Balance field.

A screen template is an ASCII file created with a text editor. The screen template files you create must reside in or under the Meridian IVR 2.0/I /u/ivr/3270 directory and must have the file name extension **.scn**. For example, if you created a screen template called **customer.scn**, it would have these paths:

```
u/ivr/3270/customer.scn
```

For each accessed field, there should be a *field descriptor* specified that governs how data is retrieved from or entered in the field. The screen template can include both data input entries as well as data output entries.

Screen template syntax

The syntax of a screen template is shown in Figure 2-11.

Figure 2-11
Screen template syntax

```
#comment
screen-name validation tag offset validation tag
field-descriptor
field-descriptor
•
•
•
key-descriptor
sleep-descriptor
```

The lines depicted as • represent additional *field-descriptor* lines. The example in Figure 2-12 illustrates a screen template file that obtains the balance from the screen shown in Figure 2-10.

Figure 2-12
Screen template for accounting application

```
#Screen template file to obtain the current balance; filename: customer.scn  
customer1,1Account  
#places balance into buffer  
0,0Balance:$
```

In Figure 2-12, the first line is a comment describing the screen template file. The *screen-name* is “customer,” the name of the screen template file without the **.scn** extension. The “1,1” represents the location of the validation tag on the screen. The row is listed first, followed by the column. “Account” is the screen validation tag.

The fourth line is the *field-descriptor* that describes an action to take. This *field descriptor* is going to find an exact match to “Balance:” and place the contents of the field into a buffer. The *field-descriptor* line has many variations, depending on what you want to do with a field. For example, the third line in Figure 2-12 could be entered as

```
2,48 — $
```

This would place the contents of the field starting at 2,48 into the next buffer. See “field-I/O” later in this section.

An explanation of each entry in the screen template follows.

#comment

The first line of the template in Figure 2-12 is a comment. The comment line is not required but is recommended to describe the purpose of the screen template.

Comments can be embedded anywhere in the screen template and start with the “#” symbol. If a comment begins a line, no non-comment fields may follow the comment. However, a comment may appear after a non-comment field, such as after the field descriptor line shown in Figure 2-12. It is good programming practice to heavily comment files.

screen-name

The first non-comment line specifies the name of the screen. This is the screen template file name without the **.scn** extension.

validation-tag offset

This entry specifies the position of the *validation-tag* by row and column. You may enter **0,0** for the *validation-tag offset* in only two cases:

- To indicate that you do not want to validate this screen (you would also need to enter a hyphen, “-”, as the validation-tag). You should only ignore the identity of a screen if you want the TRS process to perform the actions specified in the screen template regardless of what screen is actually active. For example, if you want to execute a command at the system prompt, you would not need to verify that you are at a specific screen, as long as you are sure the screen has a system prompt.

Note: In general, you should validate screens whenever possible. This ensures data for the host computer application is sent to the correct screen, and data sent back to the Meridian IVR 2.0/I application is from the correct screen.

- To tell the TRS to search the screen for the validation tag. If you do not know the exact location of the validation tag or if the location of the validation varies, you can tell the TRS to search the screen for the tag. Enter 0,0 for the offset, then enter the validation tag in the appropriate field.

validation-tag

This entry specifies the *validation-tag* used on the screen. The entry should be text that always appears in the same location every time this screen displays. For the example, “Account” is always displayed starting at location 1,1 whenever a customer’s Account Receivable screen is displayed.

At the end of the previous screen template, you may want to use a clear screen function (or execute the ENTER key 24 times using *key-descriptor* lines) so you know the starting point for information on the screen, especially if the screens you are accessing on the host computer scroll. For example, if you execute an application and it starts displaying information at the current cursor location, you need to know the current cursor location to be able to validate that screen.

Enter a hyphen, “-”, for the *validation-tag* to indicate that you do not want to validate this screen (you would also need to enter **0,0** as the *validation-tag offset*). As described previously, you should only use this method of validation if you want the TRS process to perform the action specified in the template regardless of what screen is actually active.

Note: If the screen you need to validate is a blank screen, enter this line for the items: *screen-name*, *validation-tag offset*, and *validation-tag*:

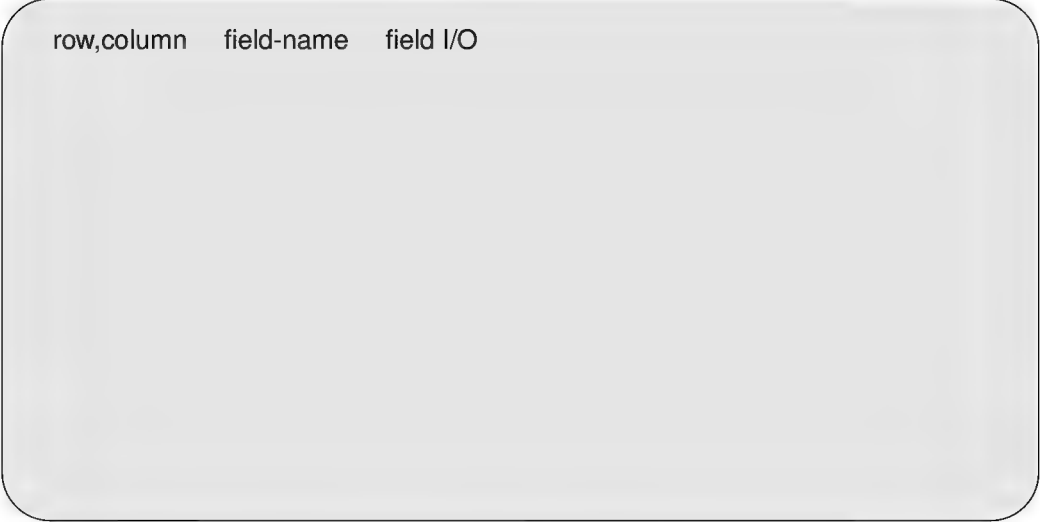
```
blank1, 1BLANKSCREEN
```

The word “blank” is a place-holder; the 1,1 and BLANKSCREEN are required.

field-descriptor

This line identifies the location and name of a field on the screen, and the action to be performed. You can enter as many *field-descriptor* lines as necessary to perform the task needed on the screen. The *field-descriptor* lines should be entered in the same order as they are accessed for the transaction. Figure 2-13 shows the syntax for *field-descriptor* lines.

Figure 2-13
The field-descriptor syntax



```
row,column  field-name  field I/O
```

Note: If the application running on the host is stream-based, meaning the screen scrolls as the user enters data and retrieves responses, you should enter the field-descriptor as follows:

```
0,0- BLANKOUT
```

In this instance, the field-descriptor will clear the TRS memory space associated with the application screen so that your application call flow will know where to retrieve the appropriate data.

row,column If the TRS is going to read information from a field on the host screen, this entry locates the field on the screen. If you know the exact location of the field the TRS is reading from, you can enter it in row, column format. If you do not know the exact location of the field the TRS is reading from, or if the location of the field varies, you can enter 0,0 and the TRS will search for a match to the name specified in *field name*. For unformatted, character-based applications, you must specify the exact location.

In VT100 transactions, writing to the terminal screen always occurs at the current cursor position. Therefore, if the field I/O action is writing text to the host screen, the TRS will write the text to the screen at the current cursor position regardless of the row,column you specify.

If you enter “0,0” and a hyphen for the **row**, **column** and **field-name** entries, the **field I/O** specified is executed at the current cursor location.

To locate field-names on the screen for reading, the offset of the upper left corner of the screen is 1,1 and the lower, right corner is 24,80.

field-name This entry identifies the field being accessed on the terminal screen. If the *field-descriptor* is 0,0 (i.e., for an exact match), the *field-name* must uniquely match text on the screen or must be a hyphen to indicate that the field is at the current cursor position. When entering the field name, keep in mind that the TRS process right-justifies all field names and removes all extra white space. If the transaction fails, this field identifies the current screen so the reset-action template can be executed.

If you enter “0,0” and a hyphen for the **row**, **column** and **field-name** entries, the **field I/O** specified is executed at the current cursor location.

To specify a specific location on the screen that does not have an associated **field-name**, use a hyphen, “-”, for the **field-name** entry. If you do, you must specify a location, such as 24,8 for the location of the action specified by the **field-I/O** entry.

field-I/O This entry indicates the action to be taken on the field. Table 2-1 explains the valid entries for this field.

Table 2-1
Valid field I/O entries

Entry	Description
*	Inputs the contents of the next input buffer (transmitted from the Meridian IVR 2.0/I application) into the field and used for COMI cells.
\$num	Outputs the contents of the field into the next output buffer. num is a number in the range 1-31 and represents the number of characters in the field TRS will put in the output buffer. If you do not assign num a value, TRS will place 31 characters into the buffer. It is used for COMO cells.
%n\$	Outputs the contents of the field into an internal variable named n, which must be a number from 1–9.
%n*	Enters the contents of internal variable n into the field.
%n\$num	Outputs the first num characters of the field into variable n.
text	Any text string to be entered in the field. If any of the special characters listed in this table are to be entered as text, enclose the entire text in quotes (e.g., "\$abc").
BLANKOUT	Clears TRS memory space associated with the host application screen before retrieving output.

Note: Place the asterisk and dollar sign characters in quotation marks if your field-descriptors require their use without their associated buffer commands.

For retrieving data into an output buffer, **\$num** indicates the number of characters to be retrieved from the field; using **\$** without a number retrieves characters until an attribute is encountered, or a maximum of 31 characters have been retrieved.

Internal variables (indicated by the **%** symbol) are used within the VT100 screens only and cannot be transmitted through the TRS gateway. You can only use internal variables to store and enter data from one screen to another in the host computer application. To send the data to the Meridian IVR 2.0/I application that called the TRS process, you need to store the data in an output buffer.

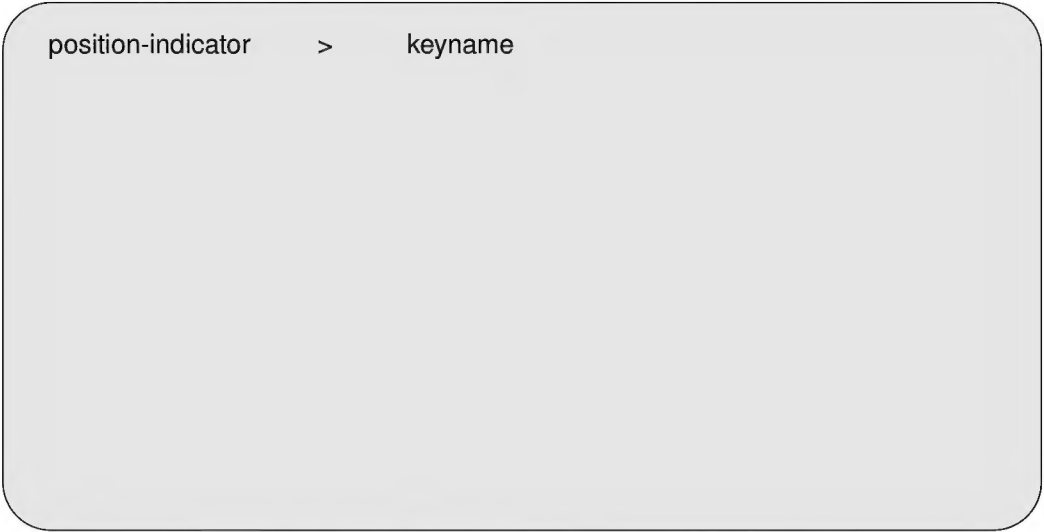
When entering a *field-descriptor*, the entries on the line should be separated by white space or tab characters. If you are entering text for the field I/O, the TRS ignores any white space you include with the value until it encounters the new line character (e.g., the value includes several words).

Note: The order of the *field-descriptor* lines in your screen template file determines how data is entered and retrieved from the screens and written into the input and output buffers of a COMO cell. Chapter 4 describes how to use Meridian IVR 2.0/I cells to retrieve data from the host computer.

key-descriptor

Identifies a key to be used with the screen. To send information to the computer for processing or to execute a command, the terminal operator presses a key on the terminal keyboard. If you do not specify a key, nothing is sent to the host and the current screen does not change. The key could be the **ENTER** key or a function key. The format of this line is similar to the *field-descriptor*. Figure 2-14 contains the format of the *key-descriptor*.

Figure 2-14
Key-descriptor line syntax

A diagram showing the syntax for a key-descriptor line. It consists of a light gray rounded rectangle containing the text 'position-indicator', a right-pointing arrow '>', and 'keyname'.

```
position-indicator    >    keyname
```

position-indicator This entry should be set to 0,0.

> This character indicates that this line contains a key.

keyname The name of the key for this screen. Table 2-2 lists the valid keys you can enter.

Table 2-2
Valid key names

ATTENTION	FORWARDWORD	PF11
BACKSPACE	HOME	PF12
BACKTAB	INSERTCHAR	PF13
BACKWORD	NEWLINE	PF14
CLEAR	PA1	PF15
CURSORDOWN	PA2	PF16
CURSORLEFT	PA3	PF17
CURSORLEFTDBL	PF1	PF18
CURSORRIGHT	PF2	PF19
CURSORRIGHTDBL	PF3	PF20
CURSORUP	PF4	PF21
DELCHAR	PF5	PF22
DUP	PF6	PF23
ENTER	PF7	PF24
ERASEEOF	PF8	RESET
ERASEINPUT	PF9	SYSREQUEST
FIELDMARK	PF10	TAB

This line may appear anywhere after the *screen-name* line (i.e., the first non-comment line). As with *field-descriptor* lines, the entries on this line must be separated by white space or tab characters.

sleep-descriptor (optional)

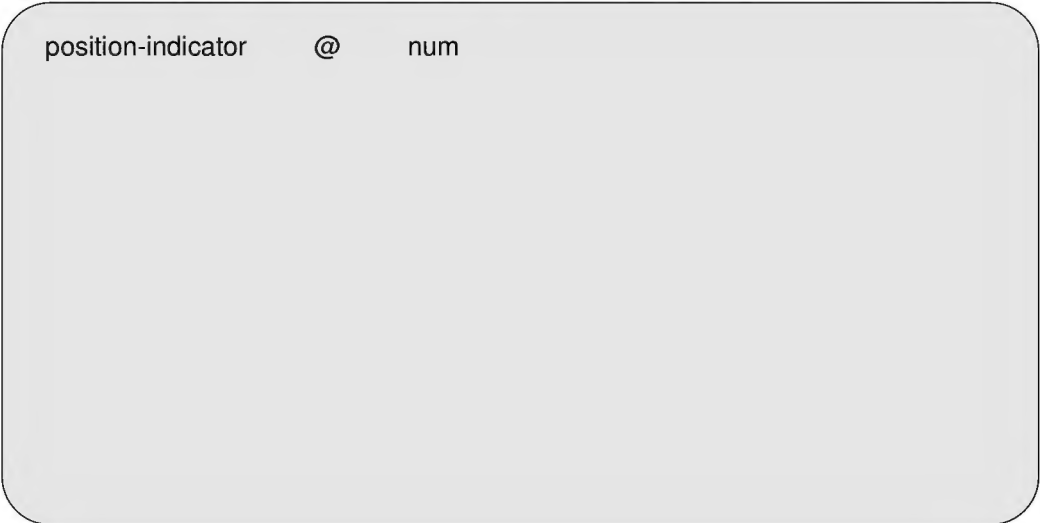
A *sleep-descriptor* causes the transaction to pause for a specified number of seconds. You can use a *sleep-descriptor* to pause the transaction for a specified period of time before or after processing a *key-descriptor*. The waiting period takes into account the time the host computer may take to process the information entered on the screen, and to move to the next screen. The transaction waits at any point where you place a *sleep-descriptor*.

Usually, a *sleep-descriptor* is placed after a *key-descriptor* to indicate that the session should wait for a specified amount of time to ensure that the next screen is ready.

Note: There is an inherent 5 second transaction time for each *sleep-descriptor* input. The total value of the *sleep-descriptor* and the inherent time must not exceed the time out value. For example, with 10 buffers, the inherent wait time is a minimum of 50 seconds. Therefore, the time out value should be greater than 50 seconds.

Each *sleep-descriptor* follows the syntax shown in Figure 2-15.

Figure 2-15
Sleep-descriptor syntax



```
position-indicator @ num
```

position-indicator This entry should be set to 0,0.

@ Identifies that this line is a *sleep-descriptor*.

num Specifies the number of seconds that the session should sleep. For example, to wait 25 seconds, you would code the sleep-descriptor as

0,0 @25

Initial-action templates

Before you can process any information on the host computer using the VT100 Gateway, you need to set the starting point for each of your terminal sessions. For example, you may want session 2 to start processing at an application's main menu, whereas session 5 should start at the system prompt.

You automatically initialize each terminal session by defining initial-action templates. Initial-action templates are action templates that specify a sequence of screen templates that position the terminal session at the desired location on the host computer whenever TRS is started up.

The initial-action template follows the same format and syntax as defined for action templates earlier in this chapter. If you must use “*” or “\$” characters in your field-descriptors, put them in quotation marks.

Instead of referencing these action templates in the COMI cell, you specify initial-action templates in an ASCII file named **trs.conf**. In addition, the session numbers defined in the **trs.conf** file must have a corresponding entry in a file named **vt100.ctl**, which assigns a device type for the session. See Chapter 3 for information on the configuration and syntax of the *trs.conf* and *vt100.ctl* files.